NASA/CR—97-206240

LSPRAY–A Lagrangian Spray Solver User's Manual

M.S. Raju
NYMA, Inc., Brook Park, Ohio

Prepared under Contract NAS3–27186

National Aeronautics and
Space Administration

Lewis Research Center

November 1997

Available from

NASA Center for Aerospace Information
800 Elkridge Landing Road
Linthicum Heights, MD 21090-2934
Price Code: A04

National Technical Information Service
5287 Port Royal Road
Springfield, VA 22100
Price Code: A04

LSPRAY - a Lagrangian Spray Solver - User's Manual

M.S. Raju
Nyma, Inc., NASA Lewis Research Center
2001 Aerospace Parkway
Brook Park, Ohio-44142

Abstract

LSPRAY is a Lagrangian spray solver developed for application with parallel computing and unstructured gas flow solvers. It is designed to be massively parallel and could easily be coupled with any existing gas-phase flow and/or Monte Carlo Probability Density Function (PDF) solvers. The solver accommodates the use of an unstructured mesh with mixed elements of either triangular, quadrilateral, and/or tetrahedral type for the gas flow grid representation. It is specifically used for fuel sprays within gas turbine combustors, but it has many other uses. The manual provides the user with the coding required to couple the spray code to any given flow code and a basic understanding of the LSPRAY code structure and the models involved in the spray formulation. The source code of LSPRAY will be available with the National Combustion Code (NCC) as a complete package.

Table of Contents

I. List of symbols

$\underline{a}_n$	outward area normal vector of the nth face
B_k	Spalding transfer number, defined in Eq. 19
C_p	specific heat, J/(Kg K)
D	turbulent diffusion coefficient, m^2/s
d	droplet diameter, m
dt	time increment, s
dt_{gl}	global time step (liquid phase), s
dt_{il}	injection time step (liquid phase), s
dt_{ml}	allowable time step (liquid phase), s
h	specific enthalpy, J/Kg
l_k	latent heat of evaporation, J/Kg
M_i	molecular weight of ith species, kg/(kg-mole)
m_k	droplet vaporization rate, Kg/s
n_k	droplet number in kth group
P	pressure, N/m^2
P_n	pressure at normal conditions, N/m^2
P_r	Prandtl number
R_u	Gas constant, J/(Kg K)
r	radial coordinate (gas-phase equations) or droplet radius (liquid-phase equations), m
S_c	Schmidt number
S_k	droplet radius squared, r_k^2, m^2
S_{ml}	liquid inter-phase source term of gas-phase equations associated with dt_{ml}
S_{mlc}	liquid inter-phase source term of the gas-phase continuity equation
S_{mle}	liquid inter-phase source term of the gas-phase energy equation
S_{mlm}	liquid inter-phase source term of the gas-phase momentum equations
S_{mls}	liquid inter-phase source term of the gas-phase species equations
T	temperature, K
t	time, s
U_k	velocity component of kth droplet group, m/s
u	velocity component, m/s
V_c	volume of the computational cell, m^3
V_k	velocity component of kth droplet group, m/s
v	velocity component, m/s
W_k	velocity component of kth droplet group, m/s
w	velocity component, m/s
$\dot{w}_j$	gas phase chemical reaction rate, 1/s
x	Cartesian coordinate
Y	gaseous species mass fraction

y Cartesian/radial coordinate
z Cartesian coordinate

greek symbols

ΔV computational cell volume, m^3
δ Dirac-delta function
ϵ_j species mass fraction at the droplet surface
λ thermal conductivity, J/(m s K)
μ dynamic viscosity, kg/(m s)
ρ density, kg/(m^3)
τ viscous stress term in Eq. 3, kg/(m s^2)
χ mole fraction
σ_l surface tension, kg/s^2
θ void fraction

superscripts

$\bar{}$ time averaging
$''$ fluctuations

subscripts

b represents conditions at boiling temperature
c cell-face center
f represents conditions associated with fuel
g global or gas-phase
k droplet group or liquid phase
l liquid phase or laminar
m conditions associated with dt_{ml}
n nth-face of the computational cell
o initial conditions or oxidizer
p grid cell or particle location
s represents conditions at the droplet surface or adjacent computational cell
i coordinate or specie indices
j specie indices
, partial differentiation with respect to the variable followed by it

II. Introduction

There are many occurrences of sprays in a variety of industrial and power applications and materials processing. A liquid spray is a two phase flow with the gas as the continuous phase and the liquid as the dispersed phase in the form of droplets or ligaments.[1] The interaction between the two phases, which are coupled through exchanges of mass, momentum, and energy, can occur in different ways at disparate time and length scales involving various thermal, mass, and fluid dynamic factors.

A number of finite-difference formulations have been advanced over the years for predicting the flow (mass and momentum) and thermal properties of a rapidly vaporizing spray. Some of the pros and cons of various formulations can be found in Refs. 1 to 3. Depending on the nature of the spray, an appropriate selection could be made from the choice of multicontinua, discrete-particle, and probabilistic spray formulations, Lagrangian or Eulerian representation for the liquid-phase equations, and different vaporization models. In this manual, we only summarize the salient aspects of the spray formulation adopted from our previous work[4-9] without attempting to provide an in-depth review on the subject of the thermal theory and fluid dynamic behavior of reacting sprays.

LSPRAY employs the multicontinua approach, which allows for resolution on a scale greater than the average spacing between two neighboring droplets.[1] An Eulerian scheme is assumed for the gas phase equations and a Lagrangian scheme is used for the liquid phase equations as it eliminates errors associated with numerical diffusion. The vaporization model of a polydisperse spray takes into account the transient effects associated with the droplet internal heating and the forced convection effects associated with droplet internal circulation and the phenomena associated with boundary layers and wakes formed in the intermediate droplet Reynolds number range.[5] The present formulation is based on a deterministic particle tracking method and on a dilute spray approximation which is applicable for flows where the droplet loading is low. Not considered in the present release of the code are the effects associated with the droplet breakup, the droplet/shock interaction, the multi-component nature of liquid spray and the phenomena associated with dense spray effects and super-critical conditions. The numerical method used could be used in both unsteady and steady state calculations.[6-9]

The success of any numerical methodology used in the study of practical combustion flows depends not only on the modeling and numerical accuracy considerations, but its applicability would be dictated mainly by the available computer memory and turnaround times afforded by the present-day computers.

Our previous work on the extension of the Monte Carlo PDF approach to spray flames showed that the computer turnaround times and memory restrictions could be reduced considerably by exploiting the recent advances in parallel computer architectures.[8] It is well known that considerable effort usually goes into generating traditional structured grid meshes for gridding up practical combustor geometries which tend to be very complex in shape and configuration. The grid generation time could be reduced considerably by making use of existing automated unstructured grid generators.[10]

With the aim of advancing the current multi-dimensional computational tools used in the design of advanced technology combustors, we have recently extended our previous work on sprays to unstructured grids following the guidelines established for the development of the National Combustion Code (NCC).[9] NCC is being developed in the form of a collaborative effort between NASA LeRC, aircraft engine manufacturers, and several other government agencies and contractors.[11] Some of the salient features of our work in Ref. 9 are summarized below:

(1) An efficient particle search algorithm was developed and implemented into the Lagrangian spray solver in order to facilitate particle movement in an unstructured grid of mixed elements.

(2) The spray solver was designed to be massively parallel in order to exploit the recent advances in parallel architectures. The spray code was rewritten in Fortran 77 with PVM calls for parallel computing.

(3) LSPRAY is currently coupled with Pratt and Whitney's CORSAIR[12] - an unstructured flow solver, and an Eulerian-based Monte Carlo probability density function solver - EUPDF,[13] which were selected to be integral components of the NCC cluster of modules. EUPDF was developed for application with sprays, combustion, unstructured grids and parallel computing.

(4) The spray solver receives the mean velocity and turbulence fields from the flow solver. The species solution supplied could be provided by either a conventional CFD solver or a Monte Carlo PDF solver depending on the choice of the solver.

(5) The spray solver computes the interphase source terms which are used by the gas phase solvers. This output could be used in both conservative as well as non-conservative finite-difference formulations of the gas phase equations.

The furnished code demonstrates the the successful methods used for coupling and parallelization of the spray to the flow code. These methods can be adopted for coupling other spray codes to Navier-Stokes solvers. Not all of the spray coding is furnished in this manual. The code that is generic to spray codes has been omitted. Only code that is unique to coupling is given. However, the first release of the generic code will be available along with NCC as a complete package.

The spray model provided several favorable results when applied to stratified-charge rotary combustion (Wankel) engines and several other confined as well as unconfined spray flames.[6-9]

III. Governing Equations for the Gas Phase

Here, we summarize the conservation equations for the gas phase in Eulerian coordinates derived for the multicontinua approach.[1] This is done for the purpose of identifying the interphase source terms arising from the exchanges of mass, momentum, and energy with the liquid phase.

The conservation of the mass leads to:

$$[\bar{\rho}V_c]_{,t} + [\bar{\rho}V_c u_i]_{,x_i} = S_{mlc} = \sum_k n_k\, m_k \tag{1}$$

For the conservation of the species, we have:

$$[\bar{\rho}V_c Y_j]_{,t} + [\bar{\rho}V_c u_i Y_j]_{,x_i} - [\bar{\rho}V_c D Y_{j,x_i}]_{,x_i} - \bar{\rho}V_c \dot{w}_j = S_{mls} = \sum_k \epsilon_j\, n_k\, m_k \tag{2}$$

where

$$\sum_j \dot{w}_j = 0 \; and \; \sum_j \epsilon_j = 1$$

For the momentum conservation, we have:

$$[\bar{\rho}V_c u_i]_{,t} + [\bar{\rho}V_c u_i u_j]_{,x_j} + [pV_c]_{,x_i} - [\theta V_c \tau_{ij}]_{,x_j} - [(1-\theta)V_c \tau_{lij}]_{,x_j} = S_{mlm} =$$

$$\sum_k n_k\, m_k\, u_{ki} - \sum_k \frac{4\pi}{3}\, \rho_k\, r_k^3\, n_k\, u_{ki,t} \tag{3}$$

where θ = the void fraction of the gas which is ratio of the equivalent volume of gas to a given volume of a gas and liquid mixture. For dilute sprays, the void fraction is assumed to be equal to one. The shear stress τ_{ij} in Eq. 3 is given by:

$$\tau_{ij} = \mu[u_{i,x_j} + u_{j,x_i}] - \frac{2}{3}\delta_{ij} u_{i,x_j}$$

For the energy conservation, we have:

$$[\bar{\rho}V_c h]_{,t} + [\bar{\rho}V_c u_i h]_{,x_i} - [\theta V_c \lambda T_{,x_i}]_{,x_i} - [(1-\theta)V_c \lambda_l T_{,x_i}]_{,x_i}$$
$$- [\theta V_c p]_{,t} = S_{mle} = \sum_k n_k\, m_k \Big(h_s - l_{k,eff}\Big) \tag{4}$$

IV. Governing Equations for the Liquid Phase

The equations of motion for each class of droplets are:

$$\frac{dx_k}{dt} = U_k \tag{5}$$

$$\frac{dy_k}{dt} = V_k \tag{6}$$

$$\frac{dz_k}{dt} = W_k \tag{7}$$

The above equations are for droplet position. For droplet velocity, we have:

$$\frac{dU_k}{dt} = \frac{3}{16}\frac{C_D \mu_{gs} Re_k}{\rho_k r_k^2}\left[U_g - U_k\right] \tag{8}$$

$$\frac{dV_k}{dt} = \frac{3}{16}\frac{C_D \mu_{gs} Re_k}{\rho_k r_k^2}\left[V_g - V_k\right] \tag{9}$$

$$\frac{dW_k}{dt} = \frac{3}{16}\frac{C_D \mu_{gs} Re_k}{\rho_k r_k^2}\left[W_g - W_k\right] \tag{10}$$

where

$$Re_k = 2\frac{r_k \rho_g}{\mu_{gs}}\left[(U_g - U_k)^2 + (V_g - V_k)^2 + (W_g - W_k)^2\right]^{1/2} \tag{11}$$

$$C_D = \frac{24}{Re_k}\left(1 + \frac{Re_k^{2/3}}{6}\right) \tag{12}$$

For droplet size, the droplet regression rate is determined from three different correlations depending upon the droplet-Reynolds-number range. When $Re_k > 20$, the regression rate is determined based on a gas-phase boundary-layer analysis[14] valid for Reynolds numbers in the intermediate range. The other two correlations valid when $Re_k \leq 20$ are taken from Clift et al.[15]

$$\frac{dS_k}{dt} = -2\frac{\mu_l}{\rho_k}\left[\frac{2}{\pi}Re_k\right]^{1/2} f(B_k) \quad if\ Re_k > 20$$

$$\frac{dS_k}{dt} = -\frac{\mu_l}{\rho_k}\left[1 + (1 + Re_k)^{1/3}\right] Re_k^{0.077} ln(1 + B_k)$$

$$if\ 1 < Re_k \leq 20 \tag{13}$$

$$\frac{dS_k}{dt} = -\frac{\mu_l}{\rho_k}\left[1 + (1 + Re_k)^{1/3}\right] ln(1 + B_k) \quad if\ \ Re_k < 1$$

where B_k is the Spalding transfer number defined in Eq. 19. The function $f(B_k)$ is obtained from the solution of Emmon's problem. The range of

validity of this function was extended in Raju and Sirignano[4] to consider the effects of droplet condensation.

The internal droplet temperature is determined based on a vortex model.[14] The governing equation for the internal droplet temperature is given by:

$$\frac{\partial T_k}{\partial t} = 17\frac{\lambda_l}{C_{pl}\rho_l r_k^2}\left[\alpha\frac{\partial^2 T_k}{\partial \alpha^2} + (1 + C(t)\alpha)\frac{\partial T_k}{\partial \alpha}\right] \tag{14}$$

where

$$C(t) = \frac{3}{17}\left[\frac{C_{pl}\rho_l}{\lambda_l}\right] r_k \frac{dr_k}{dt} \tag{15}$$

where α represents the coordinate normal to the streamsurface of a Hill's Vortex in the circulating fluid and $C(t)$ represents a nondimensional form of the droplet regression rate. The initial and boundary conditions for Eq. 14 are given by

$$t = t_{injection}, \quad T_k = T_{k,o} \tag{16}$$

$$\alpha = 0, \quad \frac{\partial T_k}{\partial \alpha} = \frac{1}{17}\left[\frac{C_{pl}\rho_l}{\lambda_l}\right] r_k^2 \frac{\partial T_k}{\partial t} \tag{17}$$

$$\alpha = 1, \frac{\partial T_k}{\partial \alpha} = -\frac{3}{32}\frac{\rho_k}{\lambda_l}\left[\frac{C_p(T_g - T_{ks})}{B_k} - l_k\right]\frac{dS_k}{dt} \tag{18}$$

where $\alpha = 0$ refers to the vortex center and $\alpha = 1$ refers to the droplet surface.

The Spalding transfer number is given by

$$B_k = \frac{C_p(T_g - T_{ks})}{l_{k,eff}} = \frac{(y_{fs} - y_f)}{(1 - y_{fs})} \tag{19}$$

$$l_{k,eff} = l_k + 4\pi\frac{\lambda_l r_k^2}{m_k}\left(\frac{\partial T_k}{\partial r}\right)_s \tag{20}$$

$$y_{fs}^{-1} = 1 + \frac{M_a}{M_f}\left(\chi_{fs}^{-1} - 1\right) \tag{21}$$

where $l_{k,eff}$ is the effective latent heat of vaporization as modified by the heat loss to the droplet interior, and M_a is the molecular weight of the gas excluding fuel vapor.

Based on the assumption that phase equilibrium exists at the droplet surface, the Clausius-Clapeyron relationship yields

$$\chi_{fs} = \frac{P_n}{P} exp\left[\frac{l_k}{R_u}\left(\frac{1}{T_b} - \frac{1}{T_{ks}}\right)\right] \tag{22}$$

In Eq. 11 the molecular viscosity is evaluated at a reference temperature using Sutherland's equation

$$\mu(T_{ref}) = 1.4637\ 10^{-6} \frac{T_{ref}^{3/2}}{T_{ref} + 120} \tag{23}$$

where

$$T_{ref} = \frac{1}{3} T_g + \frac{2}{3} T_{ks} \tag{24}$$

The droplets may evaporate, move along the wall surfaces, and/or reflect with reduced momentum upon droplet impingement with the combustor walls. In our present computations, subsequent to the droplet impingement with the walls, the droplets are assumed to flow along the wall surfaces with a velocity equal to that of the surrounding gas.

V. Details of Fuel Injection

The success of any spray model depends a great deal on the specification of the appropriate injector exit conditions. However, a discussion involving the physics of liquid atomization is beyond the scope of the this manual. In the present release of the code, the liquid fuel injection is simulated by introducing a discretized parcel of liquid mass in the form of spherical droplets at the beginning of every fuel-injection time step.

For certain cases, the fuel-injection time step, dt_{il}, needs to be determined based on the resolution permitted by the length and time scales associated with several governing parameters such as average grid spacing and average droplet spacing and velocity. However, for the case of a steady state solution, our experience showed that a time step based on the average droplet lifetime yields better convergence,[4-7] whose value typically ranges between 1 to 2 milli-seconds for the case of reacting flows.

The program facilitates fuel injection through the use of a single fuel injector comprising of different holes.[6-7] However, multiple fuel injection in a steady state calculation could be simulated by simply assigning different initial conditions for the spatial locations of the droplet groups associated with each one of the different holes. For a polydisperse spray, the program expects inputs for the number of droplet groups in a given stream and for the initial droplet locations and velocities. However, the number of droplets in a given group and their sizes could be either input directly or computed from a properly chosen function for the droplet size distribution. The specified initial inputs should be representative of the integrated averages of the experimental conditions.[8-9]

One correlation typical of those used for the droplet size distribution is taken from Ref. 16:

$$\frac{dn}{n} = 4.21\ 10^6 \left[\frac{d}{d_{32}}\right]^{3.5} e^{-16.98\left(\frac{d}{d_{32}}\right)^{0.4}} \frac{dd}{d_{32}} \tag{25}$$

where n is the total number of droplets and dn is the number of droplets in the size range between d and $d + dd$. The Sauter mean diameter, d_{32}, could be either specified or estimated from the following correlation[17]:

$$d_{32} = B_d \frac{2\pi\sigma_l}{\rho_g V_T^2} \lambda_m^* \tag{26}$$

where B_d is a constant, V_T is the average relative velocity between the liquid interface and the ambient gas, and λ_m^* is a function of the Taylor number, $(\rho_l \sigma_l^2)/(\rho_g \mu_l^2 V_T^2)$.

LSPRAY contains a subroutine, **dropdis**, for the integration of a droplet size distribution function which could easily be modified for other correlations of similar kind. The droplet size distribution obtained from the above correlation in terms of the cumulative percentage of droplet number and mass as a function of the droplet diameter is shown in Fig. 1.

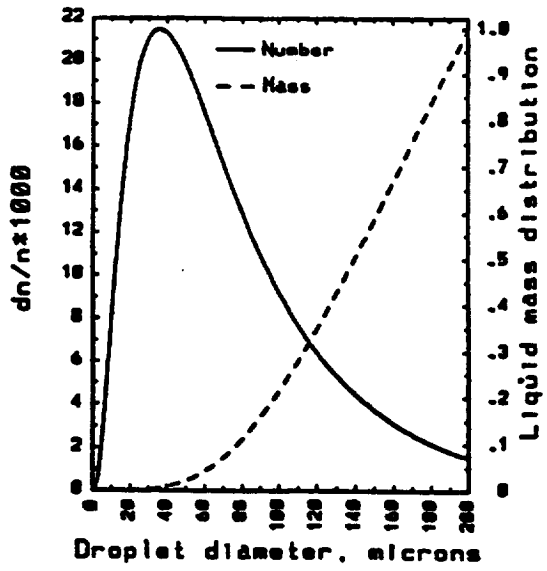

Figure 1. Droplet-size distribution.

VI. Details of the Numerical Method

In order to evaluate the initial conditions for the integration of the liquid phase equations, we need to know the gas phase properties at each particle location. But in order to evaluate the gas phase properties it is first necessary to identify the computational cell where a particle is located. It is a

trivial task to search for the computational cell of the particle location in rectangular coordinates. However, a search for the particle location becomes a complicated problem when the computational cell is no longer rectangular in the physical domain. An efficient particle search algorithm is developed and implemented into the Lagrangian spray solver in order to facilitate particle movement in an unstructured grid of mixed elements. The search is initiated in the form of a local search from the computational cell of the previous time-step as the starting point. The location of the computational cell is determined by evaluating the dot product of $\underline{x}_{pc} \cdot \underline{a}_n = |x_{pc}|\ |a_n|\ cos(\phi)$, where $\underline{x}_{pc}$ is the vector defined by the particle location to the center of the n-face of the computational cell and $\underline{a}_n$ is the outward area normal of the n-face as shown in Fig. 2, and ϕ is the angle between the two vectors.

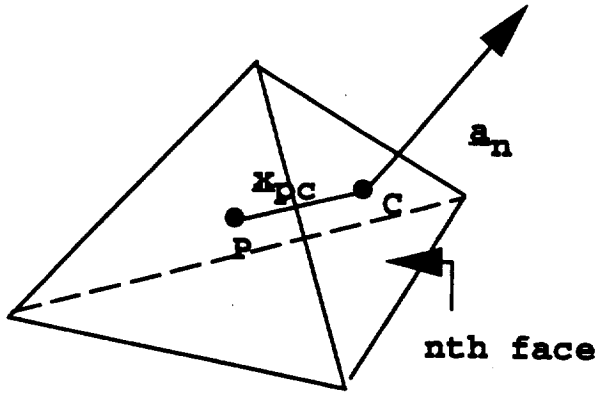

Figure 2. A vector illustration used in the particle search analysis.

A simple test for the particle location requires that the dot product be negative over each and every one of the n-faces of the computational cell. If the test fails, the particle search is carried on over to the adjacent cells of those faces over which the dot product turns out to be positive. Some of those n-faces might represent the boundaries of the computational domain while the others are the interfaces between two adjoining interior cells. The search is first carried on over to the adjacent interior cells in the direction pointed out by the positive sign of the dot products. The boundary conditions are implemented only after making sure that all the possibilities lead to a search outside of the computational boundaries. This implementation ensures against any inadvertent application of the boundary conditions before locating the correct interior cell.

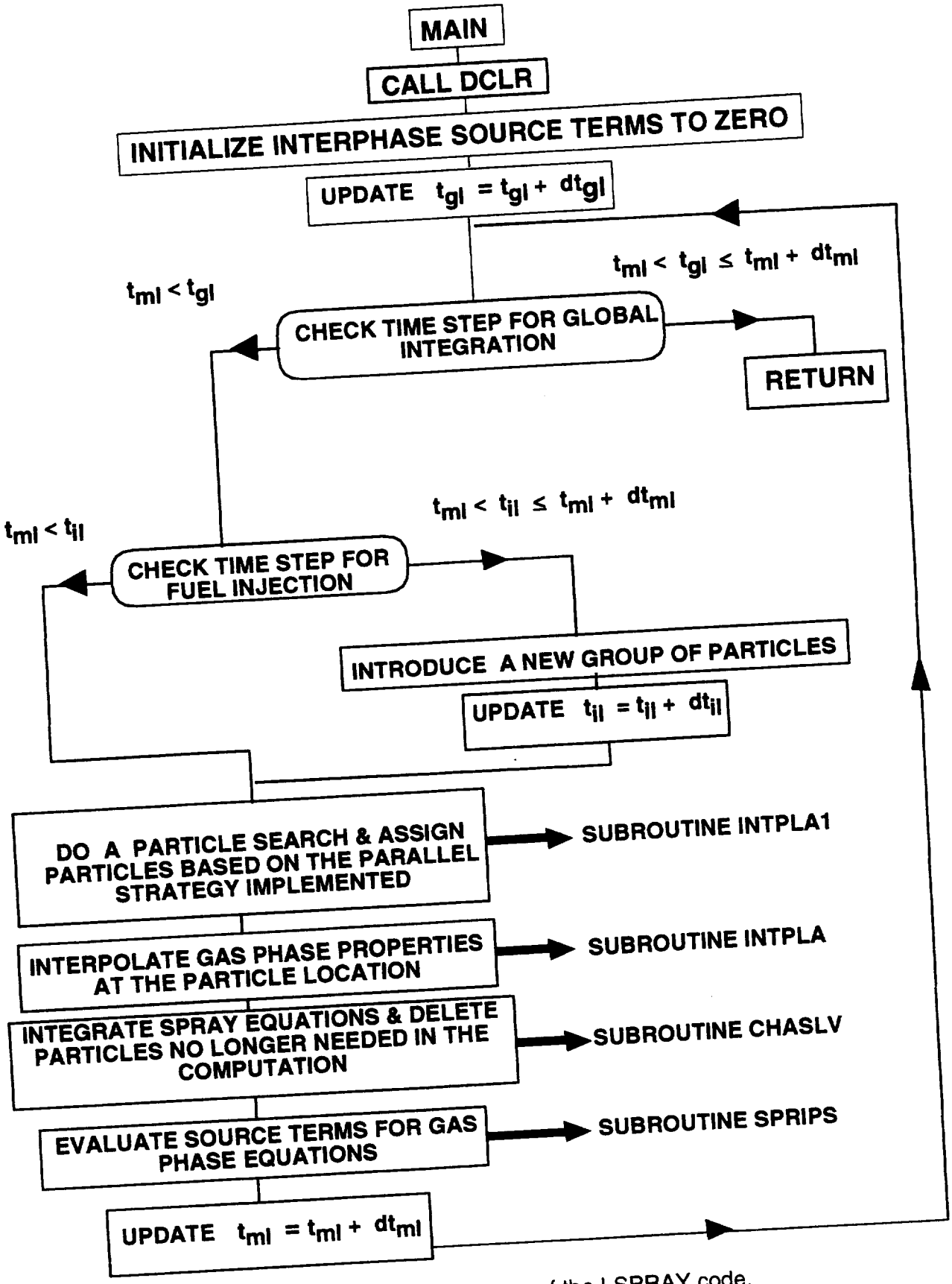

Fig. 3 The flow structure of the LSPRAY code.

After the gas phase properties at the particle location are known, the ordinary differential equations of particle position, size, and velocity are advanced by a second-order accurate Runge-Kutta method. The partial differential equations governing the droplet internal temperature distribution are integrated by an Euler method. After the liquid phase equations are solved, the interface source terms of the gas phase equations are evaluated.

VII. Program LSPRAY Flow Structure & Time-Averaging of the Interphase Source Terms of the Gas Phase Equations

The spray solver makes use of three different time steps - dt_{ml} is the allowable time step, dt_{gl} is the global time step, and dt_{il} is the fuel injection time step. dt_{ml} needs to be evaluated based on the smallest of the different time scales, which are associated with various rate controlling phenomena of a rapidly vaporizing droplet, such as those imposed by an average droplet lifetime, the local grid spacing and a relaxation time scale associated with droplet velocity among others. This restriction usually leads to a small time-step which typically has values in the neighborhood of 0.01 milli-seconds (ms). However, our experience has shown that the convergence for the steady state computations could be improved greatly by supplying the flow and EUPDF solvers with the interphase terms obtained from a time-averaging procedure, where the averaging is performed over an average lifetime of the droplets, dt_{gl}. The variable, dt_{gl}, has values in the neighborhood of 1 ms.

The averaging scheme could be explained better through the use of a flow chart shown in Fig. 3. The main spray solver is invoked by a call to **dclr** which executes the following steps:

1. It first initializes the source terms to zero.
2. Checks to see if new particles need to be introduced.
3. Advances liquid phase equations over a pre-specified time step, dt_{ml}, with calls to the following routines:
   - **intpla1** - Does a particle search and assigns particles based on the parallel strategy implemented.
   - **intpla** - Interpolates gas phase properties at the particle location.
   - **chaslv** - Advances liquid phase equations and, also, deletes any particles that are no longer needed in the computations.
   - **sprips** - Evaluates the liquid phase source term contributions, S_{ml}, for use in the gas phase equations.

4. Continues with steps (2) and (3) until the computations are completed over a global time step of dt_{gl}.
5. Returns control over to other solvers, e.g. flow or EUPDF, and supply them with source terms, S_{gl}, averaged over dt_{gl}.

The time-averaged contribution of these source terms, S_{gl}, is given by:

$$S_{gl} = \sum_{m=1}^{M} \frac{dt_{ml}}{dt_{gl}} S_{ml} \tag{27}$$

where

$$\sum_{m=1}^{M} dt_{ml} = dt_{gl} \tag{28}$$

VIII. Parallelization

There are several issues associated with the parallelization of the spray computations. The goal of the parallel implementation is to extract maximum parallelism so as to minimize the execution time for a given application on a specified number of processors.[18] Several types of overhead costs are associated with parallel implementation which include data dependency, communication, load imbalance, arithmetic, and memory overheads. The term arithmetic overhead is the extra arithmetic operations required by the parallel implementation. Memory overhead refers to the extra memory needed. Excessive memory overhead reduces the size of a problem that can be run on a given system and the other overheads result in performance degradation.[18] Any given application usually consists of several different phases that must be performed in certain sequential order. The degree of parallelism and data dependencies associated with each of the subtasks can vary widely.[18] The goal is to achieve maximum efficiency with a reasonable programming effort.[18]

In our earlier work, we discussed the parallel implementation of a spray algorithm developed for the structured grid calculations on a Cray T3D.[8] These computations were performed in conjunction with the extension of a Monte Carlo PDF method to spray flames. The parallel algorithm made use of the shared memory constructs exclusive to Cray MPP (Massively Parallel Processing) Fortran and the computations showed a reasonable degree of parallel performance when they were performed on a NASA LeRC Cray T3D with the number of processors ranging between 8 to 32.[8] Later on,

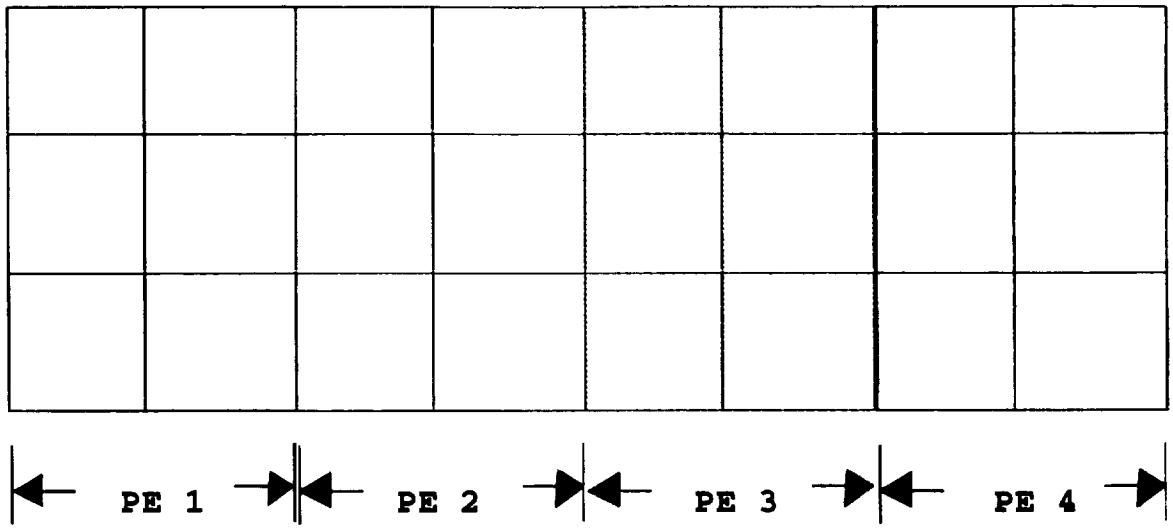

Fig. 4a An illustration of the parallelization strategy employed in the gas flow computations.

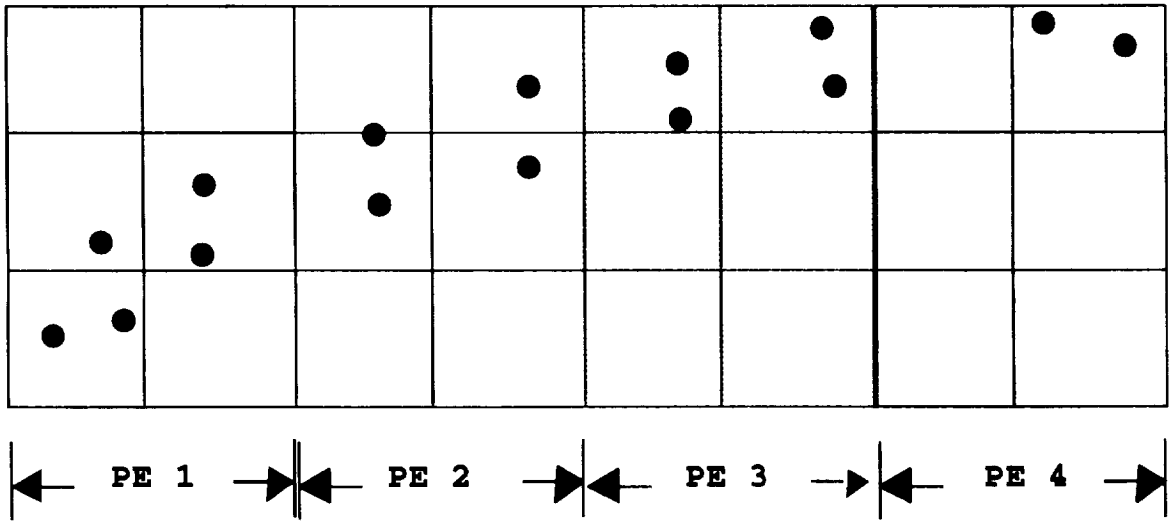

Fig. 4b An illustration of the parallelization strategy employed in the spray computations.

the extension of this method to unstructured grids and parallel computing in Fortran 77 with PVM calls was reported in Ref. 9. The Fortran 77 version offers greater computer platform independence. In this section, we only highlight some important aspects of parallelization.

In an approach, where an Eulerian scheme is employed for the gas phase computations and a Lagrangian scheme for the liquid phase computations, the gas phase computations are performed by simply dividing the domain of computation into n-parts of equal size and each part is solved by a different processor. Fig. 4a illustrates a simple example of the domain decomposition strategy adopted for the gas-phase computations where the total domain is simply divided equally amongst the available computer processing elements (PEs). In this case, we assumed the number of available PEs to be equal to four. But the Lagrangian representation makes the spray computations difficult to parallelize as spray distribution tends to be spatially very non-uniform and, also, dynamic in nature for the reasons summarized below:

(1) Most of the spray is usually confined to a small region near the atomizer location.

(2) The Lagrangian particles tend to move in and out of different parts of the computational domain processed by different PEs,

(3) Some new particles might be added to the computation at the time of fuel injection while some others might be taken out of computation either when they exit out of the computational boundaries or when they become too small after vaporization.

In order to evaluate the parallel performance of the spray computations, two different domain decomposition strategies were developed.[9] However, the present release of the code contains the one which showed better performance when the computations were performed on the LACE cluster of workstations at NASA LeRC.[9]

Fig. 4b illustrates a simple example of the domain decomposition strategy adopted for the liquid-phase computations where the corresponding gas-flow computational domain is divided into equal parts between the four available PEs. In this strategy, the Lagrangian particles are assigned to the processor of the computational grid where a particle is located. This strategy may lead to non-uniform load balancing during the integration of the liquid phase equations but is likely to result in less message passing since the inter-processor communications are limited to a single operation associated with the particle search.

IX. Details of the Coupling LSPRAY With the Flow & Monte Carlo PDF Solvers

The spray module is designed so that it could easily be coupled with any of the existing unstructured-grid flow and Monte-Carlo-PDF solvers. If geo-

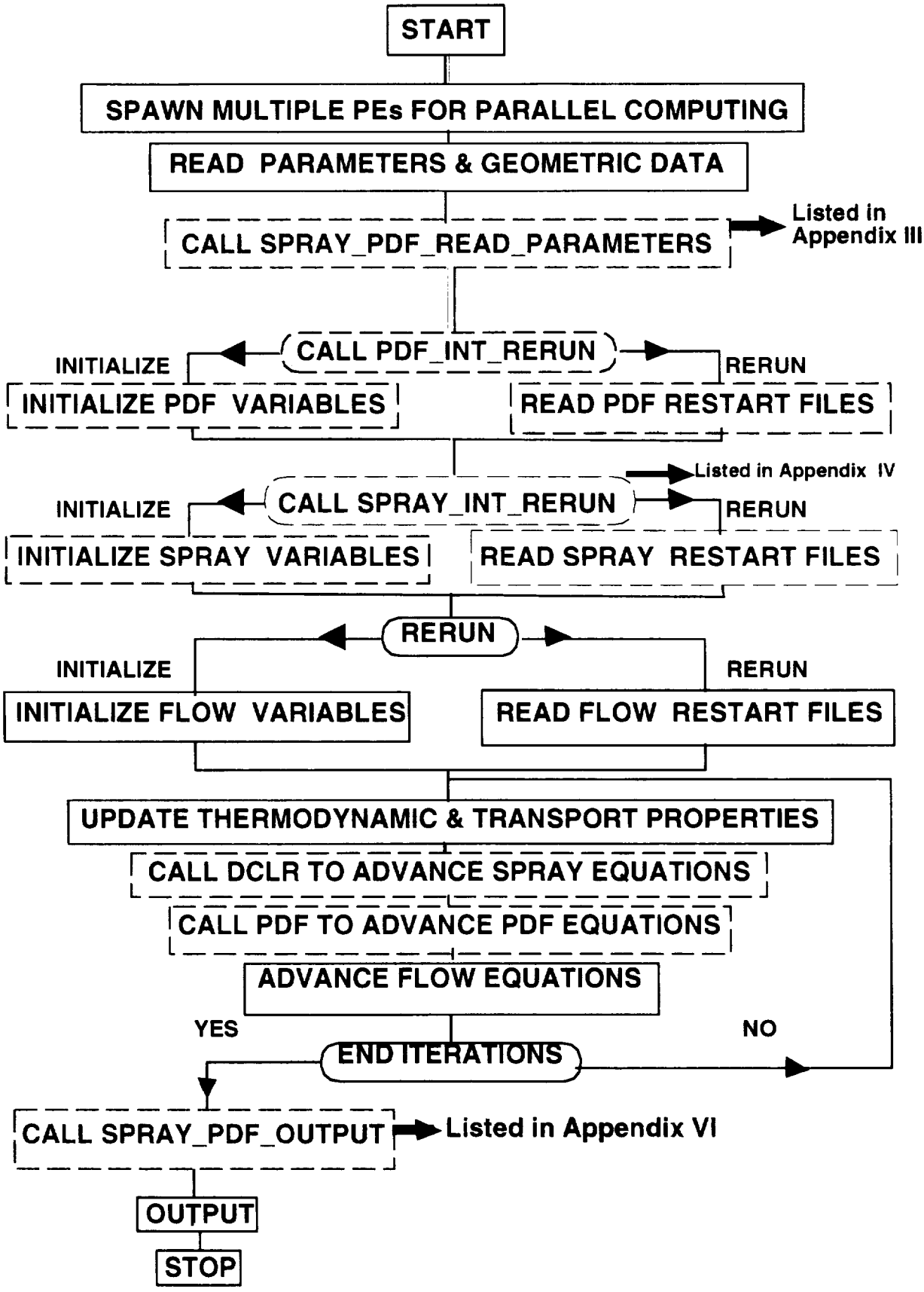

Fig. 5 The overall flow structure of the combined flow, LSPRAY, ad EUPDF solvers.

metric grid parameters - e.g. area vectors, grid connectivity, etc., were supplied separately, it could even be coupled with any of the existing structured-grid flow solvers. However, the present release of the code relies on the other modules of NCC for obtaining that information.

The structure of the spray solver is so designed that only a minimal amount of coding modifications need to be made within the flow and PDF solvers for their coupling with LSPRAY. The present version of the module relies entirely on the use of the Fortran common blocks for information exchange between the various modules. Even this reliance should entail only few changes to be made within the spray code for linkage with different solvers.

The coupling issues could be understood better through the use of a flow chart shown in Fig. 5. The chart contains several blocks - some shown in black and/or solid lines and others in color and/or dashed lines. The ones in solid blocks represent the flow chart that is typical of most flow solvers. The ones in dashed blocks represent the coupling for adding the spray and PDF solvers. The details on the PDF blocks are not provided in this report as they could be found elsewhere in a separate reference.[13] It should be borne in mind that the PDF solver could be run without the spray solver and vice-versa as they are independent.

The flow chart for a typical flow solver begins by calling several routines - some for initiating the established PVM protocol for parallel computing and the others for spawning children of the same processes so that the computations could be performed simultaneously on various PEs participating in the parallel computing environment. It is followed by a routine to read various initial parameters. The geometric data could be either read directly or created by the inclusion of appropriate calling routines needed for grid generation. Then, the initial conditions for the flow variables need to be either specified or read from the restart files if it is a rerun. The thermodynamic and transport properties are then updated before advancing the flow equations over a series of time steps until the desired number of iterations are reached. Finally, the program is terminated after writing the output data on a separate restart and standard files.

The coupling starts with the addition of a calling routine - **spray_pdf_read_parameters** - to read the spray and pdf control parameters followed by calls to the restart or initialization routines: **pdf_int_rerun** followed by **spray_int_rerun**. Then, calls to **dclr and pdf** were made in order to advance the spray and pdf equations in a sequential order before advancing the flow equations. It should be borne in mind that if the pdf solver is invoked, the thermodynamic and transport properties would be evaluated by the routines contained within the pdf solver instead of the ones contained in the flow solver. Also, Eqs. 1 to 4 - the gas phase governing equations

- need to be modified with the addition of the interphase source terms defined in Section III. Finally, a routine, **spray_pdf_output**, is included for outputting the pdf and spray data on appropriate restart and standard files.

Appendix I contains a table of all the Fortran subroutines and functions used in LSPRAY as well as the interface subroutines needed for coupling LSPRAY with a flow solver. All these routines will be released along with NCC but, however, a complete listing of all the interface subroutines is also provided in the user's manual. A brief description of all the LSPRAY routines is contained in Appendix II of the user's manual. The description pages of the LSPRAY routines along with the code listing pages of the interface routines are also given in this table.

Appendix III contains the listing of a subroutine which is used for reading some of the control and other associated parameters involving LSPRAY and EUPDF solvers. The LSPRAY initialization and restart routine is listed in Appendix IV. This routine provides a detailed account of all the input files associated with LSPRAY and gives a detailed description of how to initialize the code for the case of a swirl-stabilized confined spray flame. A sample input file is listed in Appendix V. This input applies for n-heptane fuel.

Appendix VI contains the listing of a subroutine used for writing output data from LSPRAY and EUPDF codes on separate standard and restart files.

Appendix VII contains a list of the geometric variables used by LSPRAY which are currently supplied by the flow code of NCC.

Appendix VIII contains an example of the partial listings of code initiation for coupling LSPRAY and EUPDF with a gas flow solver.

The last appendix provides an example of the summary of the CPU times taken by CORSAIR and LSPRAY for the case of a confined swirl-stabilized spray flame when the computations were performed on a LACE cluster at NASA LeRC.

X. Acknowledgements

The author would like to acknowledge Dr. W.A. Sirignano for introducing him to the subject of computational spray combustion. His sincere appreciation goes to several members of the NASA LeRC management for their past and continued support in letting him extend his work on sprays to different applications involving stratified charge Wankel engines, scalar Monte Carlo PDF, parallel computing, and unstructured grids. Many useful comments of Drs. J. Marek and T. Vanoverbeke of NASA LeRC during the preparation of this manuscript are acknowledged.

XI. References

1. Sirignano, W.A., "Fluid Dynamics of Sprays," Journal of Fluids Engineering, Vol. 115, No. 3, pp. 345-378, September 1993. Freeman Scholar Lecture, ASME Winter Annual Meeting, Anaheim, CA, November 1992.
2. Crowe, C.T., "Numerical Models for Dilute Gas-Particle Flows," Journal of Fluids, Vol. 104, pp. 297-301, 1982.
3. Faeth, G.M., "Mixing, transport, and Combustion in Sprays," Progress Energy Combustion Science, Vol. 13, pp. 293-345, 1987.
4. Raju, M.S., and Sirignano, W.A., "Spray Computations in a Centerbody Combustor," Proceedings of the 1987 ASME-JSME Thermal Engineering Joint Conference, Vol. 1, pp. 61-71, Honolulu, HI, March 1987. Also see Journal of Engineering for Gas Turbines and Power, Vol. 1, No. 4, pp. 710-718, October 1989.
5. Raju M.S., and Sirignano, W.A., "Multi-Component Spray Computations in a Modified Centerbody Combustor," Journal of Propulsion and Power, Vol. 6, No. 2, March-April 1990.
6. Raju, M.S., "AGNI-3D: A Computer Code for the Three-Dimensional Modeling of a Wankel Engine," Computers in Engine Technology, Proceedings IMechE, London, United Kingdom, pp. 27-37, 1991.
7. Raju, M.S., "Heat Transfer and Performance Characteristics of a Dual-Ignition Wankel Engine," Journal of Engines, the 1992 SAE Transactions, Section 3, pp. 466-509.
8. Raju, M.S., "Application of Scalar Monte Carlo Probability Density Function Method For Turbulent Spray Flames," Numerical Heat Transfer, Part A, Vol. 30, pp. 753-777, 1996.
9. Raju, M.S., "Combined Scalar-Monte-Carlo-PDF/CFD Computations of Spray Flames on Unstructured Grids With Parallel Computing," AIAA Paper 97-2969, 33rd AIAA/ ASME/ SAE/ ASEE Joint Propulsion Conference, July 6-10, 1997/ Seattle, WA.
10. Shang, H.M., Chen, Y.S., Liaw, P., Shih, M.H., and Wang, T.S., "Numerical Modeling of Spray Combustion With an Unstructured-Grid Method," AIAA 95-2781, 31st AIAA/ ASME/ SAE/ ASEE Joint Propulsion Conference, July 10-12, 1995/San diego, CA.
11. Liu, N.S., and Stubbs, R.M., "Preview of National Combustion Code," AIAA 97-3114, 33rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference, July 6-9, 1997/Seattle, WA.

12. Ryder, R., "CORSAIR User's Manual: Version 1.0," SID: Y965, Pratt and Whitney Engineering, United Technologies Corporation, 25 January 1993.

13. Raju, M.S., "EUPDF - an Eulerian-Based Monte Carlo Probability Density Function (PDF) Solver - User's Manual," In Preparation to be Published as a NASA Contractor's Report.

14. Tong, A.Y. and Sirignano, W.A., "Multi-component Transient Droplet Vaporization With Internal Circulation: Integral Formulation and Approximate Solution," Numerical Heat Transfer, Vol. 10, pp. 253-278, 1986.

15. Clift, R., Grace, J.R., and Weber, M.E., Bubbles, Drops, and Particles, Academic, New York, 1978.

16. El Banhawy, Y., and Whitelaw, J.H., "Calculation of the Flow Properties of a Confined Kerosene-Spray Flame," AIAA J., Vol. 18, No. 12, PP. 1503-1510, 1980.

17. Bracco, F.V., "Modelling of Engine Sprays," SAE paper 850394, 1985.

18. Ryan, J.S., and Weeratunga, S.K., "Parallel Computation of 3-D Navier-Stokes Flowfields for Supersonic Vehicles," AIAA 93-0064, 31st Aerospace Sciences Meeting and Exhibit, Reno, NV, 1993.

Appendix I

Table 1. A List of LSPRAY Fortran subroutines and functions.

Number	Name of the routine	Code released with NCC/Description page	Code page/ User's manual
1.	*blasiu*()	25	
2.	*chaslv*	25	
3.	*dclr*	25	
4.	*dropdis*()	26	
5.	*find_cntr*()	26	
6.	*find_xyzface*()	26	
7.	*intpla*	26	
8.	*intpla1*	27	
9.	*prnspr*	27	
10.	*spray_int_rerun*	27	31
11.	*spray_pdf_output*	27	49
12.	*spray_pdf_read_parameters*	27	28
13.	*sprips*	27	
14.	*sy*()	27	

Appendix II

Description of LSPRAY Fortran subroutines and functions

1. function blasiu(x):

PURPOSE: This function returns a solution based on the Blasius function used in computing the mass, momentum, and heat transfer at the droplet surface.

2. subroutine chaslv:

PURPOSE: This routine has the following functions:

(1) Integrates the liquid phase equations.

(2) Takes particles out of computation after becoming small enough due to vaporization.

3. subroutine dclr:

PURPOSE: This routine is to be called once at the beginning every global time-step, dtgl. It is primarily a controlling routine for spray computations.

This routine has the following functions:

(1) It initializes the source terms to zero.

(2) Checks to see if new particles need to be introduced.

(3) Advances liquid phase equations over an allowable or pre-specified time step, dtml, with calls to the following routines:

intpla1 - Interpolates gas phase properties at the particle locations.

chaslv - Advances liquid phase equations.

intpla - Identifies computational cells and PEs associated

with particles.

sprips - Evaluates the liquid phase contributions for use in the CFD and PDF equations.

(4) Continues with steps (2) and (3) until the computations are completed over a global time step of dtgl.

(5) Returns control over to other solvers, e.g. CFD and PDF, and also supply them with source terms averaged over dtgl.

4. subroutine dropdis(rhol,flowdum,sr,fld,smd,nofg):

PURPOSE: This routine computes droplet distribution from the following correlation:

dn/n = a*((D/D32)**alp)*exp(-b*((D/D32)**bet))*dD/D32
where a, b, alp, and bet are constants.

5. subroutine find_cntr(i,xx,yy,zz):

PURPOSE: This routine computes the x, y, and z locations of the center of the cell, i. It is used in the particle search algorithm.

6. subroutine find_xyzface(i):

PURPOSE: This routine computes x, y, and z locations of all the face centers of the element, i. It is used in the particle search algorithm.

7. subroutine intpla:

PURPOSE: This routine performs the following functions:

(1) Particle search - It really amounts to identifying the computational cell where a particle is located. In parallel computing, it also means identifying the corresponding PE associated with the partitioning of the computational domain where a particle is located.

(2) Implements appropriate boundary conditions if necessary.

(3) Reassigns the particles based on the parallel implementation stratergy employed.

8. subroutine intpla1:

PURPOSE: This routine interpolates the gas-phase properties at the particle location. In the present case, a simple zeroth order interpolation is employed.

9. subroutine prnspr:

PURPOSE: Write standard output from LSPRAY to Fortran unit 1.

10. subroutine spray_int_rerun:

PURPOSE: This routine has the following functions:

(1) Provides initial inputs to the spray computations.
(2) Restarts from previous data if it is a rerun.

11. subroutine spray_pdf_output:

PURPOSE: This routine writes output data from EUPDF & LSPRAY computations to restart and standard-output files.

12. subroutine spray_pdf_read_parameters:

PURPOSE: This routine reads controlling parameters associated with the EUPDF and LSPRAY solvers. Based on the controlling parameters read, it might invoke an initialization routine of the EUPDF solver which is needed in the thermodynamic & transport properties evaluation.

13. subroutine sprips:

PURPOSE: This routine computes the source terms arising from liquid phase contribution, of use in both CFD and Monte Carlo PDF solvers.

smlc(i) = liquid-phase contribution of Eqs. 1 and 2 of Section III.
smlmx(i), smlmy(i), smlmz(i) = liquid-phase contribution of Eq. 3 of Section III.
smle(i) = liquid-phase contribution of Eq. 4 of Section III.

14. subroutine sy(il,iu,bb,dd,aa,cc):

PURPOSE: Tri-diagonal matrix solver. It is used in the solution for the droplet internal temperature distribution.

Appendix III

A Subroutine Listing for the Read Parameters of LSPRAY and EUPDF

```
c
      subroutine spray_pdf_read_parameters
c
      include 'dcfslog.i'
      include 'dcfslog_rw.i'
c
c ------------------------------------------------
c ---
c
c PURPOSE: This routine reads controlling parameters associated
c          with the EUPDF and LSPRAY solvers. Based on the controlling
c          parameters read, it might invoke an initialization routine
c          of the EUPDF solver which is needed in the thermodynamic &
c          transport properties evaluation.
c
c FORM OF CALL: call spray_pdf_read_parameters
c
c
c ADDITIONAL I/O:
c
c    INPUT: spray_pdf_parameter_input
c
c    OUTPUT: None
c
c ------------------------------------------------
c ---
c lspray  controls turning on or off spray computations.
c lspray = .TRUE.  - turns on spray computations.
c        = .FLASE. - otherwise.
c
c ldread controls reading or not from restart files for
c        spray computations.
c ldread = .TRUE.  - restarts from previous runs.
c        = .FLASE. - starts from initial conditions.
c
c ispray_mod= This variable controls calls to the spray
c             solver. The spray solver is called once at
c             every ispray_mod times of CFD iterations.
```

```
c
c ipread = Assigned unit number for the file: liquid_input.
c idread = Assigned unit number for the file: liquid_results.
c idread2= Assigned unit number for the file: liquid_results_ini.
c idwrit = Assigned unit number for the file: liquid_results_new.
c idwrit2= Assigned unit number for the file: liquid_results_ini.
c
c ipdf controls turning on or off Monte Carlo PDF computations.
c ipdf = 0 turns off Monte Carlo PDF computations.
c      = 1 otherwise.
c
c ns serves two functions depending on whether ns has a
c    zero or non-zero value.
c ns = 0 starts the Monte Carlo PDF computations from
c        initial conditions.
c ns = a non-zero number restarts the computations from
c      a previous run. a non-zero number represents the
c      last iteration number of a previous run which is
c      used in the time-averaging scheme utilized in
c      the PDF computations.
c
c ipdf_mod  = This variable controls calls to the PDF
c               solver. The PDF solver is called once at
c               every ipdf_mod times of CFD iterations.
c
c ipdf_num = In a given cycle, the pdf solver is advanced over
c            a number of time steps given by ipdf_num.
c
c irea1 = Assigned unit number for the file: pdf_results.
c irea2 = Assigned unit number for the file: pdf_results_ave.
c iwri1 = Assigned unit number for the file: pdf_results.
c iwri2 = Assigned unit number for the file: pdf_results_ave.
c
c -------------------------------------------------
c
      open(unit=85,file='spray_pdf_parameter_input')
      read(85,*)
      read(85,*)lspray,ldread,ispray_mod
      read(85,*)
      read(85,*)ipread,idread,idwrit,idread2,idwrit2
      read(85,*)
      read(85,*)ipdf,ns,ipdf_mod,ipdf_num
```

```
      read(85,*)
      read(85,*)irea1,irea2,iwri1,iwri2
      close(unit=85)
c
c -------------------------------------------------
c ---
c
c Routine rctinp of the PDF solver provides initialization
c parameters used in the themodynamic and transport property
c evaluation as well as in the chemical kinetics scheme.
c
      if(ipdf.eq.1) then
       call rctinp
      endif
c
c -------------------------------------------------
c
      RETURN
      END
c
```

Appendix IV

A Subroutine Listing for LSPRAY Code Initialization and Restart

```
c
      subroutine spray_int_rerun
      include 'd3dpar.i'

      include 'cfsparms.i'
      include 'cfsnodes.i'

      include 'cfsxyz.i'
      include 'cfsarea.i'
      include 'cfsface.i'

      include 'cfsbc.i'
      include 'cfschar.i'
      include 'cfssym.i'
      include 'cfsadj.i'
      include 'cfstranslate.i'
      include 'cfsedge.i'
      include 'cfsperiodic.i'

      include 'parallel.i'
      include 'cfsmimd.i'

      include 'd3dcom.i'
      include 'd3dinj.i'
      include 'd3dprl.i'

      include 'dcfslog_rw.i'

      dimension fld(20),fldp(20),sr(20),srp(20)
      logical lminj,lmdis,lpinj,lpdis
c
      common/itemplvt/itemplv
c -------------------------------------------------------------
c ---
c
c
c
c PURPOSE: This routine has the following functions:
c
```

```
c      (1) Provides initial inputs to the spray computations.
c      (2) Restarts from previous data if it is a rerun.
c
c          REFERENCES:
c
c      (1) M.S. Raju, ''Heat Transfer and Performance Characteristics
c          of a Dual-Ignition Wankel Engine,'' Journal of Engines,
c          the 1992 SAE Transactions, Section 3, pp. 466-509.
c
c      (2) M.S. Raju, ''Combined Scalar-Monte-Carlo-PDF/CFD Computations
c          of Spray Flames on Unstructured Grids With Parallel Computing,''
c          AIAA paper 97-2969, the 33rd AIAA/ASME/SAE/ASEE joint propulsion
c          conference, Seattle, WA, July 06-10, 1997.
c
c FORM OF CALL: call spray_int_rerun
c
c ADDITIONAL I/O:
c
c    INPUT: liquid_input, liquid_results_ini, liquid_results.
c
c
c    OUTPUT: None
c
c
c -------------------------------------------------------
c
      itemplv=0
c
      INTS_DATA_1= 351
      INTS_DATA_2= 352
      PACKS_DATA_1= 451
c -------------------------------------------------------
c ---
c
c Initialization for parallelization.
c
c For elements whose neighboring cells are located on other
c PEs, store appropriate information on inter-processor
c communications. This information will be used in the particle
c search algorithm.
c
        ip=0
```

```
        do iwho= 1, n_neighbor_to
        i= neighbor_to(iwho)
        do n=1,number_per_processor(1,i)
        ip=ip+1
        lphi(n)  =isend_element(ip)
        enddo
        n_elem= number_per_processor(1,i)
        irc= send_data_i (iul(i),lphi, n_elem, INTS_DATA_1)
        enddo
c
        inode=nodes
        do iwho= 1, n_neighbor_from
        i= neighbor_from (iwho)
        nitems= number_per_processor(2,i)
        irc= recv_data_i (iul(i),lphi, nitems, INTS_DATA_1)
        do 594 n=1,nitems
        do 593 j=1,nfaces(inter_element(i,n))
        if(interface(i,n).eq.j) then
        bctype1(inter_element(i,n),j)=1
        element=inter_element(i,n)
        edge_id=face_to_edge(element,j)
        inode=inode+1
        edge(edge_id,1)=inode
        c1(inter_element(i,n),j)=inode
        ipr_fr_id(inode-nodes)  =i
        ile_fr_id(inode-nodes)  =lphi(n)
        endif
  593   enddo
  594   enddo
        enddo
c -------------------------------------------------------
c
c -------------------------------------------------------
c ---
c wf   = molecular weight of fuel.
c elhi = latent heat of vaporization.
c rhol = density of liquid fuel.
c cpl  = specific heat of liquid fuel.
c -------------------------------------------------------
c
c -------------------------------------------------------
c ---
```

```
c tboil = boiling temperature of fuel at normal pressure.
c conl  = thermal conductivity of lqiuid fuel.
c tdrop = initial droplet temperature.
c ugc   = universal gas constant.
c -------------------------------------------------------
c
c -------------------------------------------------------
c ---
c dtgl = Global time step. After advancing the spray
c        computations over dtgl, control is returned over
c        to other solvers such as PDF and CFD. However,
c        the allowable time step (=dtml) for advancing
c        the liquid phase equations is usually much smaller
c        than dtgl in case of steady state computations,
c        Convergence could be improved by providing a
c        spray solution based on the average life of the
c        droplets. For this reason the spray source terms
c        supplied to the CFD and PDF solvers are obtained
c        based on an averaging scheme by advancing the
c        liquid phase equations over a number of time steps
c        equaling about dtgl/dtml.
c
c dtml = allowable time step.
c -------------------------------------------------------
c
c -------------------------------------------------------
c ---
c The program facilitates fuel injection through the use of a
c single fuel injector. However, multiple fuel injection in a
c steady state calculation could be simulated by simply
c assigning different initial conditions for the spatial
c locations of the droplet groups associated with each one
c of the different holes.
c
c For a polydisperse spray, the program expects inputs
c on the number of droplet groups, droplet locations, and drop
c velocities. But droplet sizes and droplet number can be
c either input directly or could be computed by making use
c of a correlation for the droplet size distribution.
c
c
c
```

```
c lminj = .true.  - if fuel from the main injector is turned on.
c       = .false. - otherwise.
c
c lmdis = .true.  - invokes a correlation for droplet size distribution.
c       = .false. - otherwise provide inputs on droplet sizes and
c                   droplet numbers.
c
c dtil  = time step for main fuel injection.
c
c flowf = fuel flow rate, kgm/s, of main fuel injection. This input
c         is used only when lmdis = .true.
c
c smdm  = Sauter mean diameter. This input is used only when
c         lmdis = .true.
c -------------------------------------------------------
c
c-------------------------------------------------------
c ---
       write(6,*)' '
       write(6,*)' --- begins writing from subroutine spray_int_rerun of
     1spray solver ---'
       write(6,*)' '
       write(1,*)' '
       write(1,*)' --- begins writing from subroutine spray_int_rerun of
     1spray solver ---'
       write(1,*)' '
       open(unit=ipread,file='liquid_input')
       read(ipread,*)
       read(ipread,100)wf,elhi,rhol,cpl
       read(ipread,*)
       read(ipread,100)tboil,conl,tdrop,ugc
       read(ipread,*)
       read(ipread,100)dtgl,dtml
       read(ipread,*)
       read(ipread,101)lminj,lmdis,dtil,flowf,smdm
       close(unit=ipread)
c
       if(ipid.eq.1)then
       write(6,102)wf,elhi,rhol,cpl
       write(6,103)tboil,conl,tdrop,ugc
       write(6,104)dtgl,dtml
       write(6,105)lminj,lmdis,dtil,flowf,smdm
```

```
      endif
c -----------------------------------------------------
c
c -----------------------------------------------------
c ---
      fphi=16.0*atan(1.0)
      afphi=fphi/(180.0*4.0)
c -----------------------------------------------------
c
c -----------------------------------------------------
c --- redefine some parameters
c
      cyfs=elhi*wf/ugc
      cpli=cpl
      coni=conl
c -----------------------------------------------------
c
c -----------------------------------------------------
c ---
c Initialize parameters of relevance in solving the partial
c differential equation (pde) associated with the internal
c droplet temperature distribution.
c
      tonc=17.0*coni/(cpli*rhol)
      nde=13
      ndem=nde-1
      nde1=nde+2
      ndm=nde1-1
      nde11=nde1+1
      nde2=nde11+1
      dls=1.0/float(nde1-1)
      do 901 i=1,nde1
      stm(i)=float(i-1)*dls
901   continue
      dlsq=1.0/(dls*dls)
      dls=1.0/dls
c -----------------------------------------------------
c
c -----------------------------------------------------
c ---
c nr       = number of particles assigned to a given processor.
c nr_total = total number particles in the spray computations.
```

```
c
      nr=0
      nr_total=0
c -------------------------------------------------------
c
c -------------------------------------------------------------
c ----
c
c begin initialization for main fuel injection.
c
      if(lminj) then
c
c nmih = number of holes in the main fuel injector.
c nmis = number of streams per hole.
c nmip = number of droplet groups in a given stream.
c nos  = total number of groups in a polydisperse
c        spray representing the main fuel injection.
c
      nmih=1
      nmis=1
      nmip=10
      nos=nmih*nmis*nmip
c
c There are two ways to initialize the droplet sizes, sr(i),
c and droplet mass flow rates, fld(i), of a polydisperse
c spray.
c
c (1) simply input the corresponding values.
c (2) Compute from a droplet size distribution function.
c
      if(.not.lmdis) then
c
c fld(i) =  mass flow rate of ith droplet group, kgm/s
c
       fld(1)= 0.7618374184E-05
       fld(2)= 0.3031237247E-04
       fld(3)= 0.4020065171E-04
       fld(4)= 0.3439536158E-04
       fld(5)= 0.2397707249E-04
       fld(6)= 0.3062891119E-04
       fld(7)= 0.3528971865E-04
       fld(8)= 0.1766177411E-04
```

```
        fld(9)= 0.2589013457E-04
        fld(10)=0.7142747199E-04
c
c sr(i) = droplet size of the ith droplet group.
c
        sr(1)=  0.4308124971E-05
        sr(2)=  0.9601875718E-05
        sr(3)=  0.1489562419E-04
        sr(4)=  0.2019000021E-04
        sr(5)=  0.2548375051E-04
        sr(6)=  0.3254249896E-04
        sr(7)=  0.4136625284E-04
        sr(8)=  0.5018937372E-04
        sr(9)=  0.6077750004E-04
        sr(10)= 0.7665937301E-04
c
c flowf = total mass flow rate of main fuel injection.
c
       flowf =fld(1)+fld(2)+fld(3)+fld(4)+fld(5)+
     >        fld(6)+fld(7)+fld(8)+fld(9)+fld(10)
c
c flowinj = total injected fuel mass per hole per stream.
c
       flowinj=flowf*dtil/float(nmih*nmis)
c
       if(ipid.eq.1)then
       write(6,*)'flowf= ',flowf,' flowinj= ',flowinj
       write(1,*)'flowf= ',flowf,' flowinj= ',flowinj
       endif


c
c convert fld(i) into mass.
c
        fld(1)=  dtil*fld(1)
        fld(2)=  dtil*fld(2)
        fld(3)=  dtil*fld(3)
        fld(4)=  dtil*fld(4)
        fld(5)=  dtil*fld(5)
        fld(6)=  dtil*fld(6)
        fld(7)=  dtil*fld(7)
        fld(8)=  dtil*fld(8)
        fld(9)=  dtil*fld(9)
```

```
        fld(10)= dtil*fld(10)
c
        else
c
c flowinj = total injected fuel mass per hole per stream.
c
        flowinj=flowf*dtil/float(nmih*nmis)
c
        call dropdis(rhol,flowinj,sr,fld,smdm,nmip)
c
        endif
c
c determine droplet number, n_i, of ith group.
c
        andrr1=3.0*fld(1)/(fphi*rhol*(sr(1)**3))
        andrr2=3.0*fld(2)/(fphi*rhol*(sr(2)**3))
        andrr3=3.0*fld(3)/(fphi*rhol*(sr(3)**3))
        andrr4=3.0*fld(4)/(fphi*rhol*(sr(4)**3))
        andrr5=3.0*fld(5)/(fphi*rhol*(sr(5)**3))
        andrr6=3.0*fld(6)/(fphi*rhol*(sr(6)**3))
        andrr7=3.0*fld(7)/(fphi*rhol*(sr(7)**3))
        andrr8=3.0*fld(8)/(fphi*rhol*(sr(8)**3))
        andrr9=3.0*fld(9)/(fphi*rhol*(sr(9)**3))
        andrr10=3.0*fld(10)/(fphi*rhol*(sr(10)**3))
c
c store droplet sizes and droplet numbers in an array form
c for later use.
c
        do 133 iz1=1,nmih
        do 133 iz2=1,nmis
        rdrop(iz1,iz2,1)=sr(1)
        rdrop(iz1,iz2,2)=sr(2)
        rdrop(iz1,iz2,3)=sr(3)
        rdrop(iz1,iz2,4)=sr(4)
        rdrop(iz1,iz2,5)=sr(5)
        rdrop(iz1,iz2,6)=sr(6)
        rdrop(iz1,iz2,7)=sr(7)
        rdrop(iz1,iz2,8)=sr(8)
        rdrop(iz1,iz2,9)=sr(9)
        rdrop(iz1,iz2,10)=sr(10)
c
        ndrrte(iz1,iz2,1)=andrr1
```

```
         ndrrte(iz1,iz2,2)=andrr2
         ndrrte(iz1,iz2,3)=andrr3
         ndrrte(iz1,iz2,4)=andrr4
         ndrrte(iz1,iz2,5)=andrr5
         ndrrte(iz1,iz2,6)=andrr6
         ndrrte(iz1,iz2,7)=andrr7
         ndrrte(iz1,iz2,8)=andrr8
         ndrrte(iz1,iz2,9)=andrr9
         ndrrte(iz1,iz2,10)=andrr10
133      continue
c
        if(ipid.eq.1)then
         print *,' from main, ndrr= '
         print *,andrr1,andrr2,andrr3,andrr4,
     >          andrr5,andrr6,andrr7,andrr8,
     >          andrr9,andrr10
        endif
c
c In the present case, the initial particle x location
c is assumed to be the same as the center location
c of the second cell of the computational domain assigned
c to processor one.
c
c compute coordinates of cell 2 on processor one.
c
        if(ipid.eq.1)then
         ijk=2
         call find_cntr(ijk,xctr,yctr,zctr)
         xcu= xctr
         xphi(1)    =xctr
         do n=2,np
         rc = send_data_r (iul(n),xphi, 1, PACKS_DATA_1)
         enddo
        endif
        if(ipid.ne.1) then
         rc = recv_data_r (iul(1),xphi, 1, PACKS_DATA_1)
         xcu  =xphi(1)
        endif
c
c define x component of initial particle location.
c
        dxloc(1,1,1)= xcu
```

```
      dxloc(1,1,2)= xcu
      dxloc(1,1,3)= xcu
      dxloc(1,1,4)= xcu
      dxloc(1,1,5)= xcu
      dxloc(1,1,6)= xcu
      dxloc(1,1,7)= xcu
      dxloc(1,1,8)= xcu
      dxloc(1,1,9)= xcu
      dxloc(1,1,10)=xcu
c
c define y component of initial particle location.
c
      dyloc(1,1,1)= 0.6093000062E-02
      dyloc(1,1,2)= 0.5948499776E-02
      dyloc(1,1,3)= 0.5214500241E-02
      dyloc(1,1,4)= 0.4474000074E-02
      dyloc(1,1,5)= 0.4054999910E-02
      dyloc(1,1,6)= 0.3895000089E-02
      dyloc(1,1,7)= 0.4139999859E-02
      dyloc(1,1,8)= 0.5106000230E-02
      dyloc(1,1,9)= 0.4997999873E-02
      dyloc(1,1,10)=0.4944500048E-02
c
c define z component of initial particle location.
c
      dzloc(1,1,1)= 0.0
      dzloc(1,1,2)= 0.0
      dzloc(1,1,3)= 0.0
      dzloc(1,1,4)= 0.0
      dzloc(1,1,5)= 0.0
      dzloc(1,1,6)= 0.0
      dzloc(1,1,7)= 0.0
      dzloc(1,1,8)= 0.0
      dzloc(1,1,9)= 0.0
      dzloc(1,1,10)=0.0
c
c Assign approximate computational cell and processor IDS.
c
      do 132 iz1=1,nmih
      do 132 iz2=1,nmis
      do 132 iz3=1,nmip
      ieloc(iz1,iz2,iz3)=3
```

```
      iploc(iz1,iz2,iz3)=1
 132  continue
c
c define u component of drop velocity.
c
      uloc(1,1,1) = 28.15850067
      uloc(1,1,2) = 25.18750000
      uloc(1,1,3) = 21.06999969
      uloc(1,1,4) = 17.26049995
      uloc(1,1,5) = 13.75449944
      uloc(1,1,6) = 12.12549973
      uloc(1,1,7) = 11.65199947
      uloc(1,1,8) = 14.19849968
      uloc(1,1,9) = 18.00499916
      uloc(1,1,10)= 3.732000113
c
c define v component of drop velocity.
c
      vloc(1,1,1) = 34.446
      vloc(1,1,2) = 30.130
      vloc(1,1,3) = 22.646
      vloc(1,1,4) = 16.191
      vloc(1,1,5) = 11.856
      vloc(1,1,6) = 10.621
      vloc(1,1,7) = 8.212
      vloc(1,1,8) = 13.201
      vloc(1,1,9) = 11.249
      vloc(1,1,10)= 4.845
c
c define w component of drop velocity.
c
      wloc(1,1,1) = 15.246
      wloc(1,1,2) = 12.511
      wloc(1,1,3) = 10.227
      wloc(1,1,4) = 8.206
      wloc(1,1,5) = 6.760
      wloc(1,1,6) = 5.251
      wloc(1,1,7) = 5.830
      wloc(1,1,8) = 3.972
      wloc(1,1,9) = 5.255
      wloc(1,1,10)= 4.203
c
```

```
c load the above particle attributes into appropriate
c arrays  based on  the stratergy  used  for parallel
c implementaion.
c
       do 135 iz1=1,nmih
       do 135 iz2=1,nmis
       do 135 iz3=1,nmip
       if(iploc(iz1,iz2,iz3).eq.ipid)then
       nr=nr+1
       xki(nr)=dxloc(iz1,iz2,iz3)
       yki(nr)=dyloc(iz1,iz2,iz3)*(1.35-0.70*rand())
       zki(nr)=dzloc(iz1,iz2,iz3)
       uki(nr)=uloc(iz1,iz2,iz3) *(1.25-0.50*rand())
       vki(nr)=vloc(iz1,iz2,iz3) *(1.25-0.50*rand())
       wki(nr)=wloc(iz1,iz2,iz3) *(1.30-0.60*rand())
       tki(nr)=tdrop
       ski(nr)=rdrop(iz1,iz2,iz3)*rdrop(iz1,iz2,iz3)
       rki(nr)=rdrop(iz1,iz2,iz3)
       sklim(nr)=0.04*rdrop(iz1,iz2,iz3)*rdrop(iz1,iz2,iz3)
       ndrr(nr)=ndrrte(iz1,iz2,iz3)
       isen(nr)=ieloc(iz1,iz2,iz3)
       isep(nr)=iploc(iz1,iz2,iz3)
       ins(nr)=nr
       do i3=1,nde1
       vh(nr,i3)=0.0
       enddo
       vh(nr,nde11)=0.0
       smass=fphi*rhol*(rki(nr)**3)*float(ndrr(nr))/(3.0)
       vh(nr,nde2)=smass
       endif
 135   continue
c
       nr_total=nr_total+nos
c
       endif
c
c ---- end of main fuel injecton
c -------------------------------------------------------
c
c -------------------------------------------------------
c ----
c t1 =   time associated with dtgl
```

```
c tm1=   time associated with dtml
c tl1=   time associated with dtil
c
       t1 =0.0
       tm1=0.0
       tl1  = 0.0
       if(.not.lminj)tl1  = 1.0e+10
c -------------------------------------------------------------
c
c -------------------------------------------------------------
c ----
c read restart files if ldread = .true.
c
      if(ldread) then
       if(ipid.eq.1) then
       open(unit=idread2,file='liquid_results_ini')
       read(idread2,*)nr_total
       read(idread2,*)dtilte,dtmlte,t1,tl1,tm1
       read(idread2,*)iseed
       close(unit=idread2)
       dphi(1)=dtilte
       dphi(2)=dtmlte
       dphi(3)=t1
       dphi(4)=tl1
       dphi(5)=tm1
       lphi(1)=nr_total
       lphi(2)=iseed
       do n=2,np
       ns_elm=2
       irc= send_data_i (iul(n),lphi, ns_elm, INTS_DATA_2)
       ns_elm=2*5
       rc = send_data_r (iul(n),dphi, ns_elm, PACKS_DATA_1)
       enddo
       endif
       if(ipid.ne.1) then
       ns_elm=2
       irc= recv_data_i (iul(1),lphi, ns_elm, INTS_DATA_2)
       ns_elm=2*5
       rc = recv_data_r (iul(1),dphi, ns_elm, PACKS_DATA_1)
       dtilte   =dphi(1)
       dtmlte   =dphi(2)
       t1      =dphi(3)
```

```
      tl1     =dphi(4)
      tm1     =dphi(5)
      nr_total=lphi(1)
      iseed   =lphi(2)
      endif
c
      write(1,*)' begin outputting records from restart file'
      open(unit=idread,file='liquid_results',
     > access='direct',recl=136,form='unformatted')
      do ip=1,nr_total
      irecord=ip
      read(idread,rec=irecord) ndrr(ip),ins(ip),
     1 isen(ip),xki(ip),yki(ip),zki(ip),uki(ip),
     2 vki(ip),wki(ip),tki(ip),rki(ip),ski(ip),sklim(ip),
     3 (vh(ip,j),j=1,nde+4)
      if(ip.ge.1) then
      write(1,*) ip,ndrr(ip),ins(ip),
     1 isen(ip),xki(ip),yki(ip),zki(ip),uki(ip),
     2 vki(ip),wki(ip),tki(ip),rki(ip),ski(ip),sklim(ip),
     3 (vh(ip,j),j=1,nde+4),irecord
      endif
      enddo
      close(unit=idread)
      write(1,*)' end outputting records from restart file'
c
      nr=0
      do ip=1,nr_total
      ispr_el_tno=isen(ip)
      isent=mod(ispr_el_tno,nodes)
      if(isent.eq.0)isent=nodes
      isept=(ispr_el_tno-1)/nodes + 1
       if(ipid.eq.isept)then
      nr=nr+1
      isen(nr)=isent
      isep(nr)=isept
      xki(nr)  =xki(ip)
      yki(nr)  =yki(ip)
      zki(nr)  =zki(ip)
      ski(nr)  =ski(ip)
      rki(nr)  =rki(ip)
      sklim(nr)=sklim(ip)
      uki(nr)  =uki(ip)
```

```
        vki(nr)  =vki(ip)
        wki(nr)  =wki(ip)
        tki(nr)  =tki(ip)
        ins(nr)  =ins(ip)
        ndrr(nr) =ndrr(ip)
        do jk=1,nde2
        vh(nr,jk)=vh(ip,jk)
        enddo
         endif
        enddo
       endif
c
c -------------------------------------------------------
c
c -------------------------------------------------------
c ---
c
       if(ipid.eq.1) then
       write(6,*)'nr= ',nr,' nr_total= ',nr_total
       write(6,*)'ndrr= ',(ndrr(ip),ip=1,nr)
       write(6,*)'nos= ',nos
       write(6,*)'tm1= ',tm1,' dtml= ',dtml
       write(6,*)'t1= ',t1,' dtgl= ',dtgl
       write(6,*)'tl1= ',tl1,' dtil= ',dtil
       endif
c -------------------------------------------------------
       write(6,*)' '
       write(6,*)' --- ends writing from subroutine spray_int_rerun of
     1spray solver ---'
       write(6,*)' '
       write(1,*)' '
       write(1,*)' --- ends writing from subroutine spray_int_rerun of
     1spray solver ---'
       write(1,*)' '
       return
 100   format(4e16.10)
 101   format(2l2,3e16.10)
 102   format(1x,'wf= ',f5.1,' elhi= ',e11.5,
     1' rhol= ',f4.0,' cpl= ',f8.3)
 103   format(1x,'tboil= ',f5.1,' conl= ',e11.5,
     1' tdrop= ',f5.1,' ugc= ',f11.2)
 104   format(1x,'dtgl= ',e11.5,' dtml= ',e11.5)
```

```
 105   format(1x,'lminj= ',l2,' lmdis= ',l2,
     1' dtil= ',e11.5,' flowf= ',e11.5,' smdm= ',e11.5)
       end
c
```

Appendix V

An Example LSPRAY Input file

For a description of the variables used in *liquid_input* file, refer to Appendix IV.

```
wf,elhi,rhol,cpl
0.1002000000E+030.3644438200E+060.6880000000E+030.2249275500E+04
tboil,conl,tdrop,ugc
0.3715000000E+031.0333450000E-010.3140000000E+030.8314900000E+04
dtgl,dtml
0.1500000000E-020.7500000000E-05
lminj,lmdis,dtil,flowf,smdm
 T F0.1500000000E-020.3174018348E-030.1200000000E-03
```

Appendix VI

A Subroutine Listing for LSPRAY and EUPDF Data Output

```
c
      subroutine spray_pdf_output
c
      include 'cfsparms.i'
      include 'cfsdt.i'
      include 'cfspert.i'
      include 'cfsconv.i'
      include 'cfstime.i'
      include 'cfsmimd.i'

      include 'cfsarea.i'
      include 'cfsnodes.i'
      include 'cfsvars.i'
      include 'cfsprop.i'
      include 'cfsh.i'
c
c Include common blocks associated with spray and PDF computations.
c
      include 'dcfslog.i'
      include 'dcfslog_rw.i'
c
c Include common blocks associated with PDF computations.
c
      include 'p3dpar.i'
      include 'p3dcom.i'
      include 'p3dave.i'
      include 'p3dpro.i'
c
c Include common blocks associated with spray computations.
c
      include 'd3dpar.i'
      include 'd3dcom.i'
      include 'd3dinj.i'
      include 'd3dprl.i'
c
c ------------------------------------------------
c ---
c
```

```
c PURPOSE: This routine writes output data from PDF & spray
c          computations on a separate restart and standard
c          output files.
c
c FORM OF CALL: call spray_pdf_output
c
c
c ADDITIONAL I/O:
c
c    INPUTS: None.
c
c    OUTPUTS:
c
c    liquid_results_new
c    liquid_results_ini
c    spray_pdf_parameter_input
c
c ------------------------------------------------
c
c ------------------------------------------------
c ---
c
c Write spray restart files.
c
      if(lspray) then
        open(unit=idwrit,file='liquid_results_new',
     >  access='direct',recl=136,
     >  form='unformatted')
        if(ipid.eq.1) then
         open(unit=idwrit2,file='liquid_results_ini')
         write(idwrit2,*)nr_total
         call flush(idwrit2)
         write(idwrit2,*)dtil,dtml,t1,tl1,tm1
         call flush(idwrit2)
         write(idwrit2,*)iseed
         call flush(idwrit2)
         close(unit=idwrit2)
        endif
c
        INTS_DATA_2=314
        do n=1,np
        no_to_ip(n)=nr
```

```
        enddo
        do n=1,np
        if(ipid.ne.n) then
        irc= send_data_i (iul(n),no_to_ip(n), 1, INTS_DATA_2)
        endif
        enddo
        do n=1,np
        if(ipid.ne.n) then
        irc= recv_data_i (iul(n),no_fr_ip(n),  1, INTS_DATA_2)
        endif
        enddo
        no_fr_ip(ipid)=no_to_ip(ipid)
        irecordd=0
        do n=1,ipid-1
        irecordd=irecordd+no_fr_ip(n)
        enddo
c
        do ip=1,nr
         irecord=irecordd+ip
         isent=isen(ip)+(isep(ip)-1)*nodes
         write(idwrit,rec=irecord) ndrr(ip),ins(ip),
     1  isent,xki(ip),yki(ip),zki(ip),uki(ip),
     2  vki(ip),wki(ip),tki(ip),rki(ip),ski(ip),sklim(ip),
     3  (vh(ip,j),j=1,nde+4)
         call flush(idwrit)
        if(ip.ge.1) then
        write(1,*) irecord,ndrr(ip),ins(ip),
     1 isent,xki(ip),yki(ip),zki(ip),uki(ip),
     2 vki(ip),wki(ip),tki(ip),rki(ip),ski(ip),sklim(ip),
     3 (vh(ip,j),j=1,nde+4),nr,nr_total,irecord
        endif
        enddo
         close(unit=idwrit)
       endif
c
c -----------------------------------------------
c
c -----------------------------------------------
c ---
c
c Update file: spray_pdf_parameter_input.
c Also, write PDF restart files.
```

```
c
c
      if(ipdf.eq.1) then
        if(ipid.eq.1) then
        open(unit=85,file='spray_pdf_parameter_input')
        write(85,*)'lspray ldread ispray_mod'
        write(85,*)lspray,ldread,ispray_mod
        write(85,*)'ipread idread idwrit idread2 idwrit2'
        write(85,*)ipread,idread,idwrit,idread2,idwrit2
        write(85,*)'ipdf ns ipdf_mod ipdf_num'
        write(85,*)ipdf,ns,ipdf_mod,ipdf_num
        write(85,*)'irea1 irea2 iwri1 iwri2'
        write(85,*)irea1,irea2,iwri1,iwri2
        close(unit=85)
        endif
        call outpdf2(ns)
      endif
c
c -----------------------------------------------
c
c -----------------------------------------------
c ---
c
c Write output of spray computations either to unit
c one or to the screen.
c
      if(lspray) call prnspr
c
      return
      END
c
c
```

Appendix VII

A Listing of Geometric Variables Used in LSPRAY and EUPDF

```
c -------------------------------------------------------------
c ---
c
c The spray module expects the following inputs on
c the grid related information:
c
c nodes = total number of the computational elements.
c nedge = total number of faces in the computational domain.
c nfaces(i) = total number of faces of the element, i.
c
c edge(i,1) and edge(i,2) represent the adjacent elements of
c the face, i, if the face happens to be an interface between two
c elements. Otherwise edge(i,1) represents the correponding
c boundary condition identifier if the face happens to
c represent a computational boundary condition.
c
c face_to_edge(i,j) represents the face ID of the element, i,
c and the face, j.
c
c c1(i,j) provides connectivity map. c1(i,j) = adjacent element
c ID of the element, i, and the face, j, otherwise
c c1(i,j) = boundary condition identifier on any boudary.
c
c vol(i) = volume of the element, i.
c
c areax(i), areay(i), and areaz(i) are the cartesian components
c of the outward pointing area vector of the face, i.
c
c x1(i), y1(i), z1(i) are the cartesian components of the node
c one of the element, i. Similarly, x2(i), y2(i), z2(i)
c are for node 2 and so on.
c
c triangle(i) is .true. if i is a triangular element. Similarly,
c quadrilateral(i), tetrahedron(i), and wedge(i) are logical
c varibles representing other type of elements.
c
c axisymmetric is set to .true. for axisymmetric computations
c otherwise it is .false. The axisymmetric computations are
```

```
c performed by generating 3D elements from a 2D mesh with
c an arc centered around the z coordinate, z=0.0. The angle of
c the arc is defined by the variables, ARC, in radians and THETA0,
c in degrees.
c
c -------------------------------------------------------------
c
```

Appendix VIII

An Example of the Partial Listings of Code Initiation for Coupling LSPRAY and EUPDF With a Gas Flow Solver

1. The following segment shows how include calls to spray_int_rerun & pdf_int_rerun.

```
c
c -------------------------------------------------
c ---
c
c Include common blocks associated with spray and PDF computations.
c
      include 'dcfslog.i'
      include 'dcfslog_rw.i'
c
c
c Initialize Monte Carlo PDF computations.
c
      if(ipdf.eq.1) then
      call pdf_int_rerun
      endif
c
c Initialize spray computations.
c
      IF(lspray) then
      call spray_int_rerun
      endif
c
c -------------------------------------------------
```

II. The following segment shows how to include calls to DCLR & PDF.

```
c
c Include common blocks associated with spray and PDF computations.
c
      include 'dcfslog.i'
c
      double precision tbiggas, tendgas, totaltgas
```

```
c -------------------------------------------------------
c ---
c
c
c Call dclr in order to advance the spray computations
c over a time step of dtgl.
c
      if(lspray.and.mod(iteration,ispray_mod).eq.0) then
       call dclr
      endif
c
c -------------------------------------------------------
c
c -------------------------------------------------------
c ---
c
c
c Call pdf in order to advance the PDF computations
c over the next time step.
c

      if(ipdf.eq.1.and.mod(iteration,ipdf_mod).eq.0) then
       do i=1,ipdf_num
        call pdf
       enddo
      endif
c
c -------------------------------------------------------
c
```

III. The following segment shows how to include the interphase contributions to the gas phase governing equations.

```
c -------------------------------------------------
c ---
c
c Include common blocks associated with spray and PDF computations.
c
      include 'dcfslog.i'
c
c Include common blocks associated with spray the solver.
c
```

```
      include 'd3dqat.i'
c
c -----------------------------------------------
c
c -----------------------------------------------
c ---
c
c Include liquid-phase contributions to mass, momentum, species,
c and energy equations.
c
      if(lspray) then
        do i=1,nodes
          sourcem(i)=sourcem(i)+smlc(i)
          sourceu(i)=sourceu(i)+smlmx(i)
          sourcev(i)=sourcev(i)+smlmy(i)
          sourcew(i)=sourcew(i)+smlmz(i)
          sourcef(i)=sourcef(i)+smlc(i)
          sourceh(i)=sourceh(i)+smle(i)
        enddo
      endif
c
c -----------------------------------------------
```

Appendix IX

An Example Summary of CPU Times Taken By CORSAIR and LSPRAY

Table 2 summarizes the cpu times per cycle taken by CORSAIR and LSPRAY versus the number of processors used on the NASA LeRC LACE cluster. These computations refer to the case of a confined swirl-stabilized spray flame as reported in Ref. 9. The computations were performed on a grid of 2486 quadrilateral elements. It takes approximately about 1000 to 2000 cycles for the computations to reach a converged solution. These results should be considered as preliminary because, as reported in Ref. 9, the parallel performance of the spray computations was found to be reasonable on massively parallel computers like Cray T3D but poor on workstation clusters like LACE. In ref. 9, the results obtained from different decomposition strategies were also summarized. We are in the process of exploring ways to improve upon the parallel performance of LSPRAY in a workstation-cluster environment.

Table 2. Cpu time (sec) per cycle versus number of PEs on LACE Cluster.

Solver	Characteristic	Number of processors			
		2	4	8	16
CORSAIR	5 steps/cycle	3.55	1.90	1.10	0.60
LSPRAY	100 steps/cycle	0.72	1.00	2.00	6.00

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November 1997	Final Contractor Report

4. TITLE AND SUBTITLE

LSPRAY–A Lagrangian Spray Solver–User's Manual

5. FUNDING NUMBERS

WU–523–26–33–00
NAS3–27186

6. AUTHOR(S)

M.S. Raju

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NYMA, Inc.
2001 Aerospace Parkway
Brook Park, Ohio 44142

8. PERFORMING ORGANIZATION REPORT NUMBER

E–10974

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135–3191

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR—97-206240

11. SUPPLEMENTARY NOTES

Project Manager, Sol Gorland, Turbomachinery and Propulsion Systems Division, NASA Lewis Research Center, organization code 5830, (216) 977–7561.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Categories: 07, 20, 34, 61, and 64 Distribution: Nonstandard

This publication is available from the NASA Center for AeroSpace Information, (301) 621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

LSPRAY is a Lagrangian spray solver developed for application with parallel computing and unstructured gas flow solvers. It is designed to be massively parallel and could easily be coupled with any existing gas-phase flow and/or Monte Carlo Probability Density Function (PDF) solvers. The solver accommodates the use of an unstructured mesh with mixed elements of either triangular, quadrilateral, and/or tetrahedral type for the gas flow grid representation. It is specifically used for fuel sprays within gas turbine combustors, but it has many other uses. The manual provides the user with the coding required to couple the spray code to any given flow code and a basic understanding of the LSPRAY code structure and the models involved in the spray formulation. The source code of LSPRAY will be available with the National Combustion Code (NCC) as a complete package.

14. SUBJECT TERMS

Computational spray combustion; Sprays; Two-phase flows; CFD; Combustion

15. NUMBER OF PAGES

64

16. PRICE CODE

A04

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	